UNIT – 1

1. Define Distributed System & discuss its characteristics. Give examples for Distributed Systems.

Distributed system is the one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

Characteristics:

1. **Concurrency:** Concurrent programs execution is the norm. I can do my work on my computer while you do your work on yours, sharing resources such as web pages or files when necessary. Resources can be added or removed easily.
2. **No global Clock:** Close coordination often depends on a shared idea of the time at which the programs' actions occur. But it turns out that there are limits to the accuracy with which the computers in a network can synchronize their clocks – there is no single global notion of the correct time.
3. **Independent Failure:** Faults in the network result in the isolation of the computers that are connected to it, but that doesn't mean that they stop running.

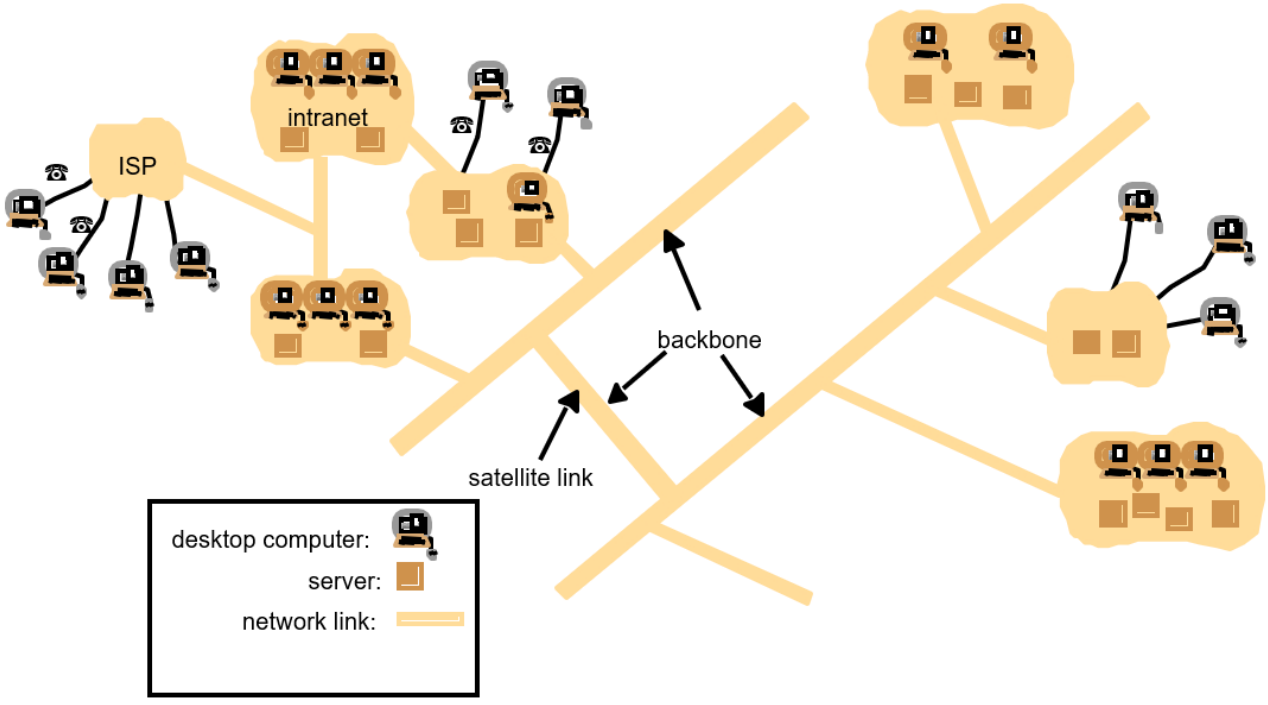
Typical examples of Distributed systems are,

1. The Internet
2. Intranets
3. Mobile and Ubiquitous computing.

The Internet

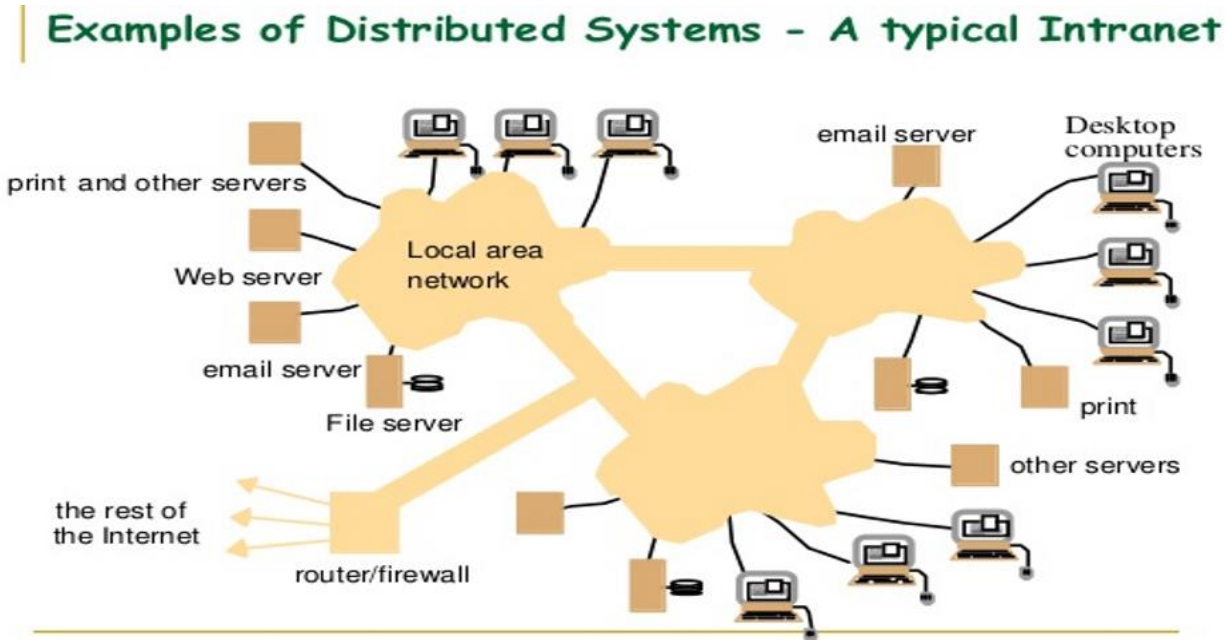
- Internet is a very large distributed system. It enables users, wherever they are, to make use of services like www, email, file transfer. The set of services is open-ended.

Refer figure below…



Intranet

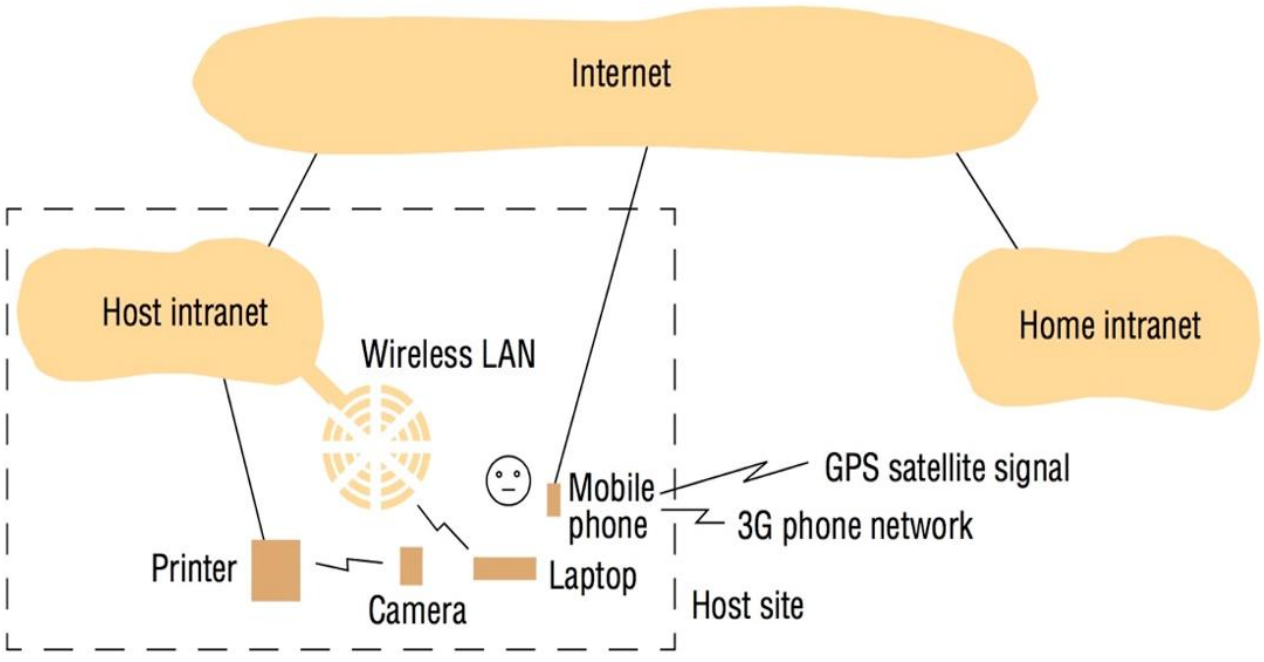
- An intranet is a portion of the internet that is separately administered and has a boundary that can be configured to enforce local security policies.
- It may be composed of several LANs linked by backbone connections.
- The n/w configuration of a particular intranet is the responsibility of the organization that administers it.
- An intranet is connected to the Internet via router, which allows the users to use the services available in the Internet.
- Firewall is used to protect intranet by preventing unauthorized messages leaving or entering.



- Some organizations do not wish to connect their internal networks to the Internet at all.
- E.g. police and other security and law enforcement agencies are likely to have at least some internal networks that are isolated from outside world.
- These organizations can be connected to Internet to avail the services by dispensing with the firewall.
- The main issues arising in the design of components for use in intranets are,
    1. File services are needed to enable users to share data
    2. Firewalls should ensure legitimate access to services
    3. Cost of installation and support should be minimum

Mobile and Ubiquitous computing

- Integration of portable computing devices like Laptops, smartphones, handheld devices, pagers, digital cameras, smart watches, devices embedded in appliances like refrigerators, washing machines, cars etc. with the distributed systems became possible because of the technological advances in **device miniaturization** and **wireless networking**.
- These devices can be connected to each other conveniently in different places, makes mobile computing possible.
- In mobile computing, users who are away from home intranet, are still allowed to access resources via the devices they carry.
- Ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users physical environments, including home, office, and others.
- The term ubiquitous is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed.
- The presence of computers everywhere is useful only when they can communicate with one another.
- E.g. it would be convenient for users to control their washing machine and hi-fi system using "Universal remote control" device at home.
- The mobile user can get benefit from computers that are everywhere.
- Ubiquitous computing could benefit users while they remain in a single environment such as the home, office, or hospital.
- Figure below shows a user who is visiting a host organization. The users home intranet and the host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.



2. List the challenges in distributed systems. Explain in detail any two of them.

The challenges faced in distributed systems are as follows:

- Heterogeneity
- openness
- Security
- Scalability
- Failure Handling
- Concurrency
- Transparency

Heterogeneity

- The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks.
- Heterogeneity (that is, variety and difference) applies to all of the following:
  - Networks
  - Computer hardware
  - Operating systems
  - programming languages
  - Implementations by different developers

openness

- The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.
- The openness of distributed system determined primarily by the degree to which new resource sharing devices can be added.
- openness cannot be achieved unless the specification and documentation of the Key software interfaces of the components of a system are made available to software developers. In a word, the key interfaces are *published*.

Security

- Many of the information in distributed systems have a high intrinsic value to their users.
- Their security is therefore of considerable importance.
- Security for information resources has three components:
  - confidentiality (protection against disclosure to unauthorized individuals),
  - integrity (protection against alteration or corruption), and
  - availability (protection against interference with the means to access the resources).

For example:

1. A doctor might request access to hospital patient data or send additions to that data.
2. In electronic commerce and banking, users send their credit card numbers across the Internet.

Scalability

- A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.
- The number of computers and servers in the Internet has increased dramatically.

Failure Handling

- When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation.
- Failures in a distributed system are partial – that is, some components fail while others continue to function.
- The following are techniques for dealing with failures,
  - Detecting failures
  - Masking failures
  - Tolerating failures
  - Recovery from failure
  - Redundancy

Concurrency

- Both services and applications provide resources that can be shared by clients in a distributed system.
- Thus, there is a possibility that several clients will attempt to access a shared resource at the same time.
- For example, a data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time.
- The process that manages a shared resource could take one client request at a time. But that approach limits throughput.
- Therefore, services and applications generally allow multiple client requests to be processed concurrently.

Transparency

- Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components.

- ***Access transparency*** enables local and remote resources to be accessed using identical operations.
- ***Location transparency*** enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
- ***Concurrency transparency*** enables several processes to operate concurrently using shared resources without interference between them.
- ***Replication transparency*** enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
- ***Failure transparency*** enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- ***Mobility transparency*** allows the movement of resources and clients within a system without affecting the operation of users or programs.
- ***Performance transparency*** allows the system to be reconfigured to improve performance as loads vary.
- ***Scaling transparency*** allows the system and applications to expand in scale without change to the system structure or the application algorithms.

3. Define Architecture Model. Mention its goal & explain the following with an example.

i. Mobile Code
ii. Mobile Agent
iii. Proxy Server & Cache
iv. Peer-Peer Network

An architectural model of a distributed system is concerned with the placement of its parts and the relationships between them.

Examples:

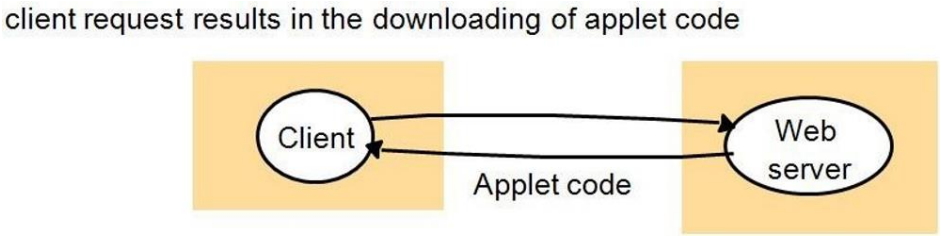
- Client-server
- Peer-to-peer

---

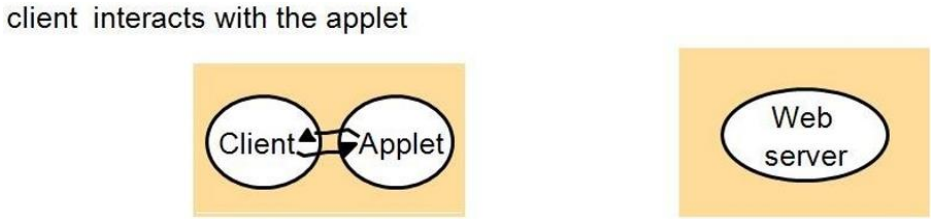
i. Mobile Code

It is used to refer the code that can be sent from one computer to another and run at the destination.

Example: java applets



Step 1: The user running a browser selects a link to an applets whose code is stored on a web server. The code is downloaded to the browser and runs there.



Step 2: Client interacts with the applet.

Advantages:

- Remote invocations are replaced by local ones.
- Good interactive response.
- Does not suffer from the delays.

Disadvantages:

- Security threat to the local resources in the destination computer.

---

ii. Mobile Agent

- Mobile agent is a running program that travels from one computer to another carrying out a task to someone's behalf, such as collecting information, eventually returning with the results. (**e.g.** Google form)
- Mobile agent is a complete program (including both code & data) that can work independently.
- Mobile agent can invoke local resources/data.

Advantages:

- Reduce communication cost and time by replacing remote invocation with local ones.

Disadvantages:

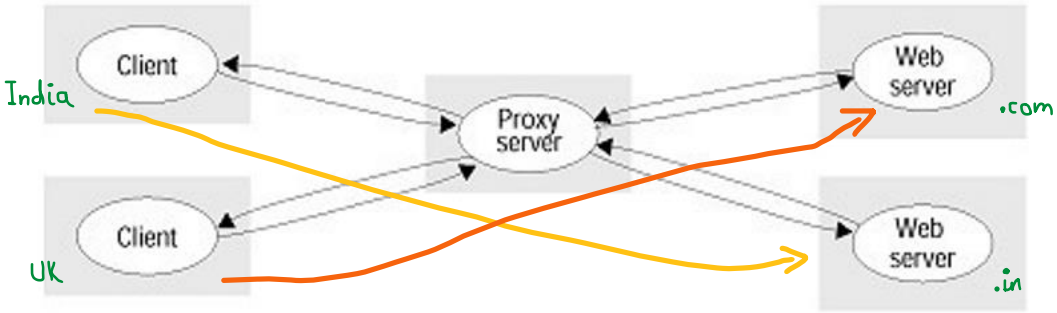
- Limited applicability
- Security threat of the visited sites resources

---

iii. Proxy Server & Cache

Proxy Servers

- Proxy servers are used to increase availability and performance of the services by reducing the load on the network and web-server.
- Proxy server provides copies (replications) of resources which are managed by other server.



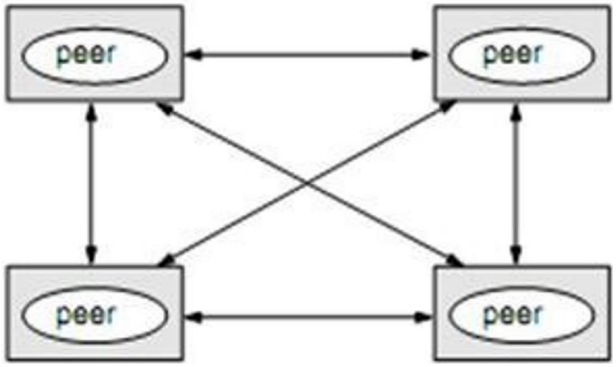
Cache

- A store of recently used data objects that is closer to the client process than those remote objects.
- When an object is needed by a client process the caching service checks the cache and supplies the objects from there in case of an up-to-date copy is available.
- **Purpose:** is to increase performance and availability by avoiding frequent accesses to remote servers.

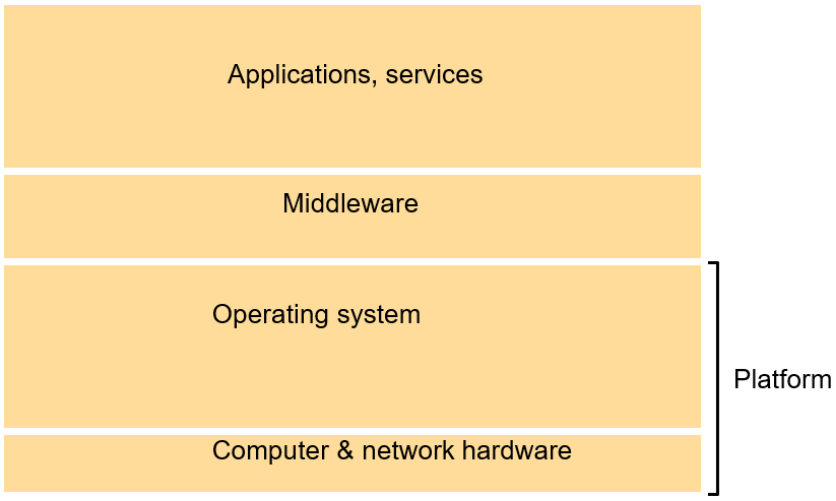
---

iv. Peer-Peer network

- All processes(objects) play similar roles without distinction between client or servers.
- A large number of data objects are shared; any individual computers hold only a small part of the application database.
- It distributes shared resources widely.
- It share computing and communication loads.



4. Discuss the Software Layers of distributed system architectural model.



Software architecture refers to services offered and requested between processes located in the same or different computers.

- structuring of software as layers or modules
- Service layers

Distributed service

- One or more server processes
- Client processes

Platform - the lowest level hardware and software layers

- Examples: Intel x86/Windows, Intel x86/Solaris, PowerPC/MAC OS, Intel x86/Linux
- Lowest level layers that provide services to other higher layers

Middleware

- masks heterogeneity & provides a convenient programming model
- Provides useful building blocks:
  Remote method invocation, communication between a group of processes, notification of events, partitioning, placement and retrieval of data or objects, replication, transmission of multimedia data in real time

---

5. Describe the interaction model of distributed system.

- Multiple server processes may cooperate with one another to provide a service;
- the examples include, the Domain Name System, which partitions and replicates its data at servers throughout the Internet,
- and Sun's Network Information Service, which keeps replicated copies of password files at several servers in a local area network.

Factors affecting interacting processes are,

- Communication performance
- No global notion of time

Communication over computer network has the following performance characteristics,

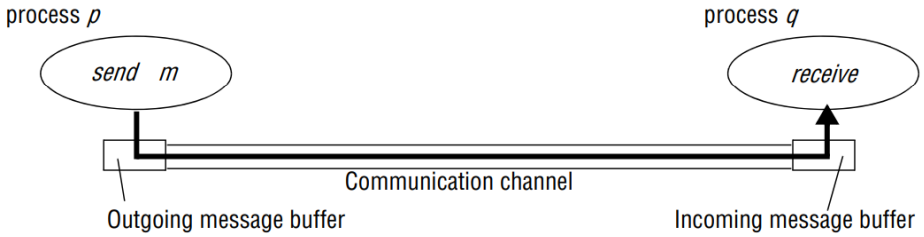
- Latency (Time taken by first string of bits to reach its destination + delay in accessing n/w+ time taken by os communication services at both ends)
- Bandwidth (Amount of data that can be sent in unit time)
- Jitter (Variation in packet delivery)

6. Describe the failure model of distributed system.

- The failure model defines the ways in which failure may occur in order to provide an understanding of the effects of failures.
- In a distributed system both processes and communication channels may fail.
- Failure model defines the types of failure,
  - Omission failure
  - Arbitrary Failure
  - Timing Failure

Omission Failures: The faults classified as *omission failures* refer to cases when a process or communication channel fails to perform actions that it is supposed to do.

Communication omission failures: Consider the communication primitives *send* and *receive*. A process *p* performs a *send* by inserting the message *m* in its outgoing message buffer. The communication channel transports *m* to *q*'s incoming message buffer. Process *q* performs a *receive* by taking *m* from its incoming message buffer and delivering it (see Figure below). The outgoing and incoming message buffers are typically provided by the operating system.

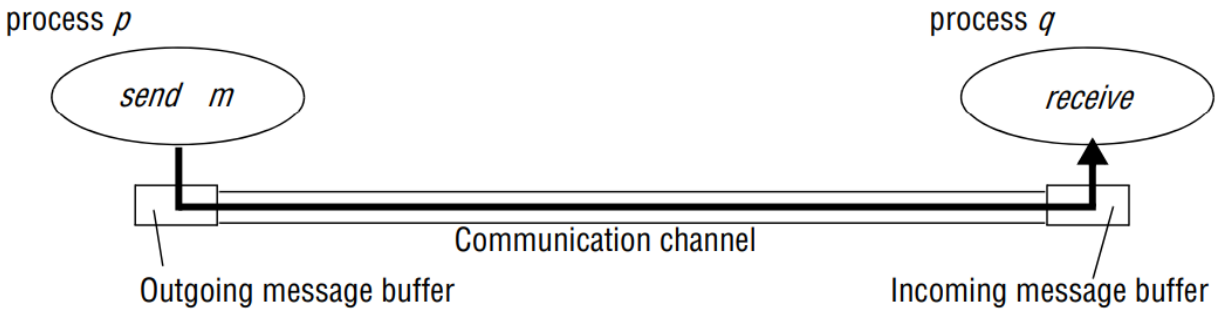


The communication channel produces an omission failure if it does not transport a message from *p*'s outgoing message buffer to *q*'s incoming message buffer. This is known as 'dropping messages' and is generally caused by lack of buffer space at the receiver or at an intervening gateway, or by a network transmission error, detected by a checksum carried with the message data.

The loss of messages between the sending process and the outgoing message buffer as *send-omission failures*, to loss of messages between the incoming message buffer and the receiving process as *receive-omission failures*, and to loss of messages in between as *channel omission failures*. The omission failures are classified together with arbitrary failures in Figure below…

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send* operation but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times or commit omissions; a process may stop or take an incorrect step.

Arbitrary Failures: The term *arbitrary failure* is used to describe the worst possible failure semantics, in which any type of error may occur. For example, a process may set wrong values in its data items, or it may return a wrong value in response to an invocation.
An arbitrary failure of a process is one in which it arbitrarily omits intended processing steps or takes unintended processing steps. Arbitrary failures in processes cannot be detected by seeing whether the process responds to invocations, because it might arbitrarily omit to reply.



Communication channels can suffer from arbitrary failures; for example, message contents may be corrupted, nonexistent messages may be delivered or real messages may be delivered more than once. Arbitrary failures of communication channels are rare because the communication software is able to recognize them and reject the faulty messages. For example, checksums are used to detect corrupted messages, and message sequence numbers can be used to detect nonexistent and duplicated messages.

Timing Failures:
Timing failures are applicable in synchronous distributed systems where time limits are set on process execution time, message delivery time and clock drift rate.

Timing failures are listed in Figure below…

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Any one of these failures may result in responses being unavailable to clients within a specified time interval.

7. **Summarize the following design requirements for Distributed Architectures**
    i. **Performance Issues**
    ii. **Quality of Service**
    iii. **Use of Caching and Replication**
    iv. **Dependability Issues**

i. Performance Issues

Responsiveness

The speed of a remote invocation depends on:

- The load and performance of the server and network
- Delays in all the software components (client and server operation systems and middleware, code of the process that implements the service)
- Transfer of data is slow

Throughput

- Rate at which computational work is done
- Fairness

Balancing of Computational Loads

- Applets remove load from the server
- Use several computers to host a single service

---

ii. Quality of Service

Reliability & Security

- ability of a system to perform and maintain its function in every circumstance

Performance

- Ability to meet Timeliness guarantees

Adaptability

- the ability of a system to adapt itself efficiently and fast to changed circumstances

---

iii. Use of Caching and Replication

- Cached copies of resources should be kept up-to-date when the resource at a server is updated.
- A variety of cache-coherency protocols are used to suit different applications.

Web Caching Protocol

- Web browsers and proxy servers cache responses to client requests from web servers
- The cache-consistency protocol can provide browsers with fresh copies of the resources held by the web server, but for performance reasons the freshness condition can be relaxed.
- A browser or proxy can validate a datum with the server.
- Web servers assign approximate expiry times to their resources
- The expiry time of the resource and the current time at the server are attached to the response.

iv. Dependability Issues

1) **Correctness**
2) **Fault Tolerance:** Dependable applications should continue to function correctly in the presence of faults in hardware, software, and networks.

Reliability is achieved through Redundancy

- Multiple computers – multiple communication paths
- Several replicas of a data item
- But redundancy is costlier.

3) **Security**

- Safety (Confidentiality) – absence of incorrect behavior
- Integrity – absence of improper system alteration
- Availability – readiness for correct service

UNIT – 2

1. Explain the characteristics of IPC.

The characteristics of IPC are as follows:

1. **Synchronous and asynchronous communication**
   - In the synchronous form, both send and receive are blocking operations.
     Eg: Continuous Chatting
   - In the asynchronous form, the use of the send operation is non-blocking and the receive operation can have blocking and non-blocking variants.
     Eg: WhatsApp

2. **Message destinations**
   - A local port is a message destination within a computer, specified as an integer.
   - A port has an exactly one receiver but can have many senders.

3. **Reliability**
   - A reliable communication is defined in terms of validity and integrity.
   - A point-to-point message service is described as reliable if messages are guaranteed to be delivered despite a reasonable number of packets being dropped or lost.
   - For integrity, messages must arrive uncorrupted and without duplication.

4. **Ordering:** Some applications require that messages to be delivered in sender order.

---

2. Compare & Contrast between Synchronous & Asynchronous communication in the context of IPC.

A queue is associated with each message destination. Sending processes cause messages to be added to remote queues and receiving processes remove messages from local queues. Communication between the sending and receiving processes may be either synchronous or asynchronous.

Synchronous:

In the synchronous form of communication, the sending and receiving processes synchronize at every message. In this case, both send and receive are blocking operations. Whenever a send is issued the sending process (or thread) is blocked until the corresponding receive is issued. Whenever a receive is issued by a process (or thread), it blocks until a message arrives.

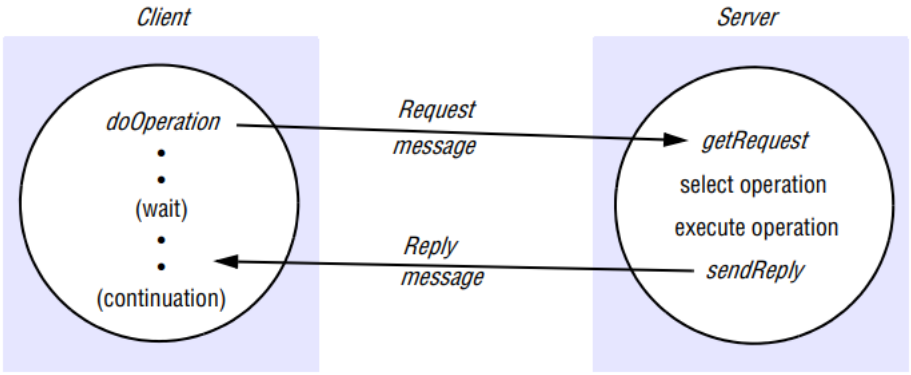
Asynchronous:

In the asynchronous form of communication, the use of the send operation is nonblocking in that the sending process is allowed to proceed as soon as the message has been copied to a local buffer, and the transmission of the message proceeds in parallel with the sending process. The receive operation can have blocking and non-blocking variants. In the non-blocking variant, the receiving process proceeds with its program after issuing a receive operation, which provides a buffer to be filled in the background, but it must separately receive notification that its buffer has been filled, by polling or interrupt.

3. Analyze the failure model of Request/Reply protocol in client-server Communication using UDP.

Request-reply communication is synchronous because the client process blocks until the reply arrives from the server.

The protocol we describe here is based on a trio of communication primitives, *doOperation*, *getRequest* and *sendReply*, as shown in Figure below...



This request-reply protocol matches requests to replies. It may be designed to provide certain delivery guarantees. If UDP datagrams are used, the delivery guarantees must be provided by the request-reply protocol, which may use the server reply message as an acknowledgement of the client request message.

- *doOperation* method is used by clients to invoke remote operations.
- The *doOperation* method sends a request message to the server whose Internet address and port are specified in the remote reference given as an argument. After sending the request message, *doOperation* invokes *receive* to get a reply message, from which it extracts the result and returns it to the caller. The caller of *doOperation* is blocked until the server performs the requested operation and transmits a reply message to the client process.
- *getRequest* is used by a server process to acquire service requests
- When the server has invoked the specified operation, it then uses *sendReply* to send the reply message to the client. When the reply message is received by the client the original *doOperation* is unblocked and execution of the client program continues.

4. **Discuss issues relating to datagram communication.**

Pg: 168 (150)txtbook

5. **Explain Characteristics and issues related to stream communication.**

Pg: 171 (153)txtbook

6. Define marshalling and unmarshalling. Explain CORBA CDR with an example

Marshalling

The process of taking a collection of data items and assembling them into a form suitable for transmission in a message.

Unmarshalling

The process of disassembling a collection of data on arrival to produce an equivalent collection of data items at the destination.

CORBA Common Data Representation (CDR)

CORBA CDR is the external data representation defined with CORBA 2.0.
It consists 15 primitive types:

- Short (16 bit)
- Long (32 bit)
- Unsigned short
- Unsigned long
- Float (32 bit)
- Double (64 bit)
- Char
- Boolean (TRUE, FALSE)
- Octet (8 bit)
- Any (can represent any basic or constructed type)

Composite type are shown in this Figure…

Type	*Representation*
sequence	length (unsigned long) followed by elements in order
string	length (unsigned long) followed by characters in order (can also can have wide characters)
array	array elements in order (no length specified because it is fixed)
struct	in the order of declaration of the components
enumerated	unsigned long (the values are specified by the order declared)
union	type tag followed by the selected member

Figure below shows a message in CORBA CDR that contains the three fields of a struct whose respective types are string, string, and unsigned long.
example: struct with value {‘Smith’, ‘London’, 1934}

index in sequence of bytes	←4 bytes→	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*‘Smith’*
8–11	"h___"	
12–15	6	*length of string*
16–19	"Lond"	*‘London’*
20–23	"on__"	
24–27	1934	*unsigned long*

7. Explain Java object serialization with an example.

- In Java RMI, both object and primitive data values may be passed as arguments and results of method invocation.
- An object is an instance of a Java class.

Example, the Java class equivalent to the Person struct

```
Public class Person implements Serializable {
        Private String name;
        Private String place;
        Private int year;
        Public Person(String aName ,String  aPlace, int aYear) {
        name = aName;
        place = aPlace;
        year = aYear;
        }
        //followed by methods for accessing the instance variables
}
```

The above class states that it implements the Serializable interface, which has no methods. Stating that a class implements the Serializable interface (which is provided in the java.io package) has the effect of allowing its instances to be serialized.

To serialize an object, its class information is written out, followed by the types and names of its instance variables. If the instance variables belong to new classes, then their class information must also be written out, followed by the types and names of their instance variables. This recursive procedure continues until the class information and types and names of the instance variables of all of the necessary classes have been written out. Each class is given a handle, and no class is written more than once to the stream of bytes (the handles being written instead where necessary).

The serialized form is shown below:

Serialized values				*Explanation*
Person	8-byte version number		h0	*class name, version number*
3	int year	java.lang.String name	java.lang.String place	*number, type and name of instance variables*
1934	5 Smith	6 London	h1	*values of instance variables*

8. **Define Marshalling. Construct a marshalled form that represents an organization with instance variable values {'KLSGIT', 'BELGAUM', 1979, 590008} by using CORBA-CDR & Java Serialization.**

9. Explain communication between distributed objects by means of RMI.

- The object model: OOP, Java or C++, review
- Distributed objects: the object model is very appropriate for distributed systems
- The distributed object model: extensions of the basic object model for distributed object implementation
- The design issues of RMI: local once-or-nothing invocation semantics vs. remote invocation semantics – similarities or differences
- The implementation issues: mapping the middleware to lower-layer facilities
- Distributed garbage collection issues

Ppt – unit 2 part 2 – slide 9 to 15?

10. Explain remote and local invocation with the neat diagrams.

11. With a neat diagram explain the role of Proxy & Skeleton in RMI

12. Explain the fundamental concepts of the distributed object model.

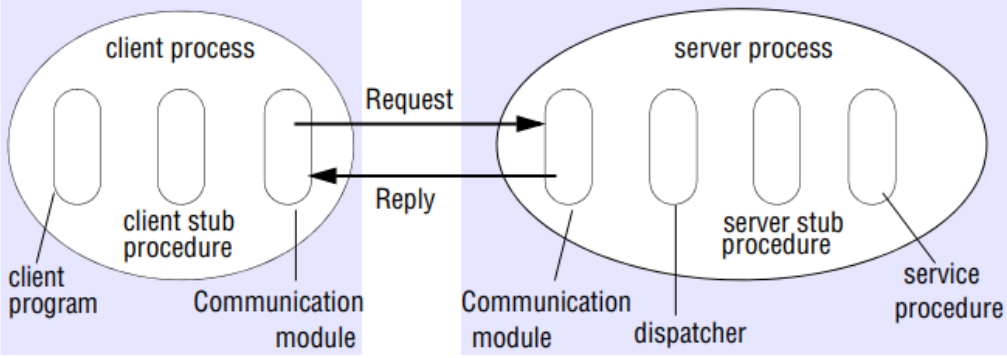
13. Discuss RMI invocation semantics and tabulate failure handling mechanism for each.

14.Define RPC and with neat diagram explain its implementation

Remote Procedure Call is a technique for building distributed systems. Basically, it allows a program on one machine to call a subroutine on another machine without knowing that it is remote.

OR

- Remote Procedure Call (RPC) is similar to RMI
- Client program calls a procedure that runs in another program running in a server process
- Servers may be clients of other servers to allow chain of RPC's
- Server Process defines Service Interface (contains the list of procedures)
- RPC chooses at least once or at most once invocation semantics.
- RPC is implemented over Request-Reply Protocol



Implementation

- The software components required to implement RPC are shown in the figure.
- The client calls the client stub. The call is a local procedure call, with parameters pushed onto the stack in the normal way.
- The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called Marshalling.
- The client's local operating system sends the message from the client machine to the server machine.
- The local operating system on the server machine passes the incoming packets to the server stub.
- The server stub unpacks the parameters from the message. Unpacking the parameters is called Unmarshalling.
- Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.

15.Explain HTTP request and reply message format.

HTTP request message

method	*URL or pathname*	*HTTP version*	*headers*	*message body*
GET	http://www.dcs.qmul.ac.uk/index.html	HTTP/ 1.1		

Method: This specifies the request type, such as GET, POST, PUT, DELETE, etc.

URI (Uniform Resource Identifier): This specifies the resource being requested, such as a web page or image.

HTTP version: This specifies the version of HTTP being used, such as HTTP/1.1.

Headers: This includes additional information about the request, such as the client's preferred language or the type of data being sent in the request body.

Body: This includes any data being sent with the request, such as form data or a JSON payload.

HTTP reply message

HTTP version	*status code*	*reason*	*headers*	*message body*
HTTP/1.1	200	OK		resource data

HTTP version: This specifies the version of HTTP being used, such as HTTP/1.1.

Status code: This is a 3-digit numerical code indicating the outcome of the request. 200 OK, 404 Not Found, 403 forbidden, 401 Unauthorized are common status codes.

Reason phrase: This is a brief, human-readable description of the status code.

Headers: This includes additional information about the response, such as the type of data being sent in the response body or the server's software version.

Body: This includes any data being sent with the response, such as the requested web page or an error message.

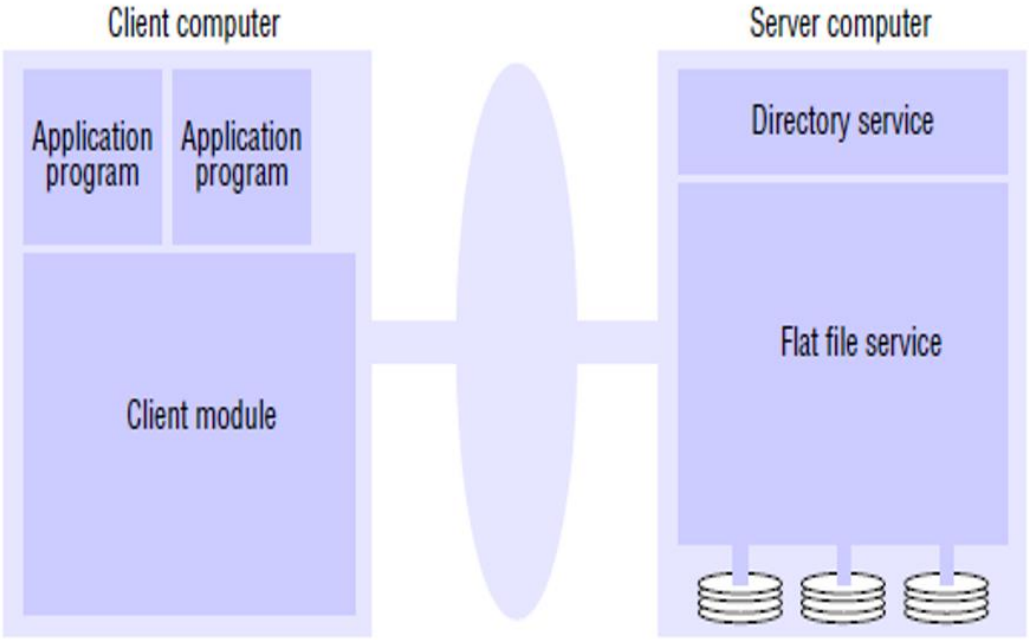
UNIT – 3

DFS

1. **Discuss model architecture of distributed file system and its components.**

2. **With a neat diagram explain the components of file service architecture in brief w.r.t. following;**
   i. **Flat File Service**

File service architecture



- The flat file service is concerned with implementing operations on the contents of files.
- *Unique file identifiers (UFIDs)* are used to refer to files in all requests for flat file service operations.
- UFIDs are long sequences of bits chosen so that each file has a UFID that is unique among all of the files.
- When the flat file service receives a request to create a file, it generates a new UFID for it and returns the UFID to the requester.

Flat File Service interface

Read(FileId, i, n) → *Data* — throws *BadPosition*	If $1 \le i \le Length(File)$: Reads a sequence of up to *n* items from a file starting at item *i* and returns it in *Data*.
Write(FileId, i, Data) — throws *BadPosition*	If $1 \le i \le Length(File)+1$: Writes a sequence of *Data* to a file, starting at item *i*, extending the file if necessary.
Create() → *FileId*	Creates a new file of length 0 and delivers a UFID for it.
Delete(FileId)	Removes the file from the file store.
GetAttributes(FileId) → *Attr*	Returns the file attributes for the file.
SetAttributes(FileId, Attr)	Sets the file attributes (only those attributes that are not shaded in Figure 12.3).

ii. Directory Service

- The directory service provides a mapping between text names for files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to the directory service.
- The directory service provides the functions needed to generate directories, to add new file names to directories and to obtain UFIDs from directories.
- Directory files are stored in files of the flat file service.

Directory service interface

Lookup(Dir, Name) → *FileId* — throws *NotFound*	Locates the text name in the directory and returns the relevant UFID. If *Name* is not in the directory, throws an exception.
AddName(Dir, Name, FileId) — throws *NameDuplicate*	If *Name* is not in the directory, adds *(Name, File)* to the directory and updates the file's attribute record. If *Name* is already in the directory, throws an exception.
UnName(Dir, Name) — throws *NotFound*	If *Name* is in the directory, removes the entry containing *Name* from the directory. If *Name* is not in the directory, throws an exception.
GetNames(Dir, Pattern) → *NameSeq*	Returns all the text names in the directory that match the regular expression *Pattern*.

---

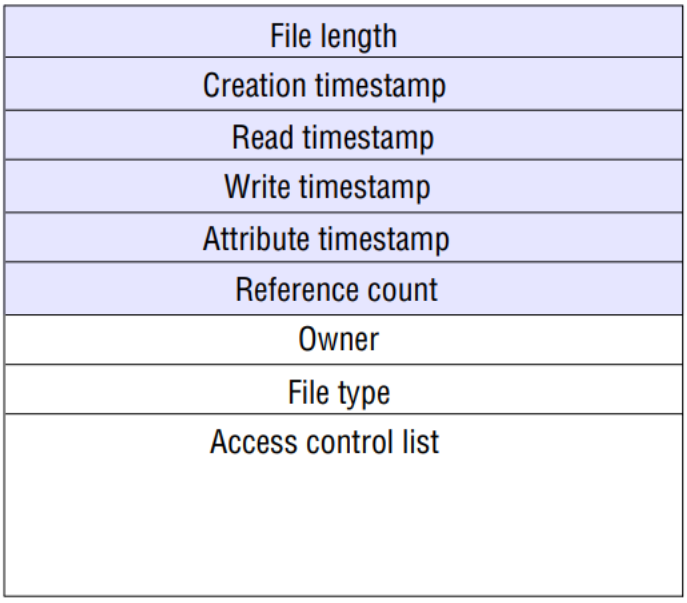
iii. Client Module

- A client module runs in each client computer, integrating and extending the operations of the flat file service and the directory service under a single application programming interface that is available to user-level programs in client computers.
- The client module also holds information about the network locations of the flat file server and directory server processes.
- Achieves satisfactory performance through the implementation of a cache of recently used file blocks at the client.

3. List out file system modules.

Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	performs disk I/O and buffering

4. Sketch the file attributes and record structure.



5. List out the transparencies in file system.

1. **Access Transparency:** Client programs should be unaware of the distribution of files. A single set of operations is provided for access to local and remote files. Programs written to operate on local files are able to access files without modification.
2. **Location Transparency:** Client programs should see a uniform file namespace. Files or groups of files may be relocated without changing their pathnames, and user programs see the same namespace wherever they are executed.
3. **Mobility Transparency:** Neither client programs nor system administration tables in client nodes need to be changed when files are moved. This allows file mobility. Files, or more commonly, sets or volumes of files may be moved, either by system administrators or automatically.
4. **Performance Transparency:** Client programs should continue to perform satisfactorily while the load on the service varies within a specified range.
5. **Scaling Transparency:** The service can be expanded by incremental growth to deal with a wide range of loads and network sizes.

---

6. List the directory service operation.

Directory service • The directory service provides a mapping between *text names* for files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to the directory service. The directory service provides the functions needed to generate directories, to add new file names to directories and to obtain UFIDs from directories. It is a client of the flat file service; its directory files are stored in files of the flat file service. When a hierarchic file-naming scheme is adopted, as in UNIX, directories hold references to other directories.

7. Describe the characteristics of file system

8. Discuss the distributed file system design requirements.

1. ***Transparency:*** Some aspects of Distributed system are hidden from user.
   - **Access:** Client programs can be unaware of distribution of files. Same set of operations are provided for access to remote as well as local files.
   - **Location:** Client program should see a uniform name space.
   - **Mobility:** Client programs need not change their tables when files are moved to any other location.
   - **Performance:** Client program should continue to perform satisfactorily while the load on the service varies.
   - **Scaling:** The service can be expanded to deal with wide range of load and network size.

2. ***Concurrent file updates:*** Changes to the file by one client should not interfere with the operation of other clients simultaneously accessing or changing the same file. (Concurrency Control)

3. ***File replication:*** A file may be represented by several copies of its contents at different locations.
   Benefits:
   1. Load balancing to enhance the scalability of the service.
   2. Enhances the fault tolerance.

4. ***Hardware and OS heterogeneity:*** The service interfaces should be defined so that client and server software can be implemented for different operating systems and computers.

5. ***Fault tolerance:*** Avoiding failure
   - To cope with transient communication failures, the design can be based on at-most-once invocation semantics.

6. ***Consistency:*** Maintaining the consistency between multiple copies of file.

7. ***Security:*** In distributed file systems, there is a need to authenticate client requests so that access control at the server is based on correct user identities and to protect the contents of request and reply messages.
   - Digital signatures and encryption of secret data is used.

8. **Efficiency:** A distributed file service should offer facilities that are of at least the same power and generality as those found in conventional file systems and should achieve a comparable level of performance.

SECURITY

1. Write the steps of RSA Algorithm. Illustrate with an example given P=3 & Q=11.

1. Choose two large prime numbers, P and Q (each greater than 10^{100}), and form
   $N = P \times Q$
   $Z = (P-1) \times (Q-1)$
2. For d choose any number that is relatively prime with Z (that is, such that d has no common factors with Z).

We illustrate the computations involved using small integer values for P and Q:

$P = 13, Q = 17 \rightarrow N = 221, Z = 192$

$d = 5$

3. To find e solve the equation:
   $e \times d = 1 \ mod \ Z$

That is, $e \times d$ is the smallest element divisible by d in the series $Z+1, 2Z+1, 3Z+1, \ldots$.

$e \times d = 1 \ mod \ 192 \ = 1, 193, 385, \ldots$

385 is divisible by d

$e = 385/5 = 77$

2. Analyze the following uses of Cryptography with suitable scenarios.

i. Secrecy and integrity

- Cryptography is used to maintain the secrecy and integrity of information whenever it is exposed to potential attacks.
- It maintains the secrecy of the encrypted message as long as the decryption key is not compromised.
- It also maintains the integrity of the encrypted information, provided that some redundant information such as a checksum is included and checked.

- **Scenario 1**. Secret communication with a shared secret key: Alice wishes to send some information secretly to Bob. Alice and Bob share a secret key *KAB.*

  1. Alice uses *KAB and an agreed encryption function E(KAB, M) to encrypt and send* any number of messages {*Mi}KAB to Bob.*

  2. Bob decrypts the encrypted messages using the corresponding decryption function *D(KAB, M).*

- **Issues:**
  - ***Key distribution****: How can Alice send a shared key KAB to Bob securely?*
  - ***Freshness of communication****: How does Bob know that any {Mi} isn't a copy of an earlier encrypted message from Alice that was captured by Mallory and replayed later?*

ii. Authentication

- Cryptography is used in support of mechanisms for authenticating communication between pairs of principals.
- Key is known only to two parties.

Scenario 2:

Authenticated communication with a server

- Bob is a file server;
- Sara is an authentication service.
- Sara shares secret key K_A with Alice and secret key K_B with Bob.

1. Alice sends an (unencrypted) message to Sara stating her identity and requesting a *ticket for access to Bob.*
2. Sara sends a response to Alice. $\{\{Ticket\}_{K_B}, K_{AB}\}_{K_A}$. It is encrypted in K_A and consists of a ticket (to be sent to Bob with each request for file access) encrypted in K_B and a new secret key K_{AB}.
3. Alice uses K_A to decrypt the response.
4. Alice sends Bob a request R to access a file: $\{Ticket\}_{K_B}$, Alice, R.
5. The ticket is actually $\{K_{AB}, Alice\}_{K_B}$. Bob uses K_B to decrypt it, checks that Alice's name matches and then uses K_{AB} to encrypt responses to Alice.

Scenario 3.

Authenticated communication with public keys

- Bob has a public/private key pair $<KB_{pub}, KB_{priv}>$

1. Alice obtains a certificate that was signed by a trusted authority stating Bob's public key Kb_{pub}.
2. Alice creates a new shared key KAB, encrypts it using Kbpub using a public-key algorithm and sends the result to Bob.
3. Bob uses the corresponding private key KB_{priv} to decrypt it.

   (If they want to be sure that the message hasn't been tampered with, Alice can add an agreed value to it and Bob can check it.)

3. **Discuss asymmetric (public/private key pair-based) cryptography technique and how it can be used in supporting security in distributed systems.**

When a public/private key pair is used, one-way functions are exploited in another way. The feasibility of a public-key scheme was proposed as a cryptographic method that eliminates the need for trust between communicating parties. The basis for all public-key schemes is the existence of a trap-door function. A trap-door function is a one-way function with a secret exit- it is easy to compute in one direction but infeasible to compute its reverse unless the secret is known.

The pair of keys needed for asymmetric algorithms are derived from a common root. The derivation of the pair of keys from the root is a one-way function. In the case of the RSA algorithm, the large numbers are multiplied together, this computation takes only a few seconds, even for very large primes used. The resulting product, N, is computationally infeasible to derive the original multiplicands.

One of the pair of keys is used for encryption. For RSA the encryption procedure obscures the plaintext by treating each block as a binary number and raising it to the power of key, modulo N. The resulting number is the corresponding ciphertext block. The size of N and at least one of the pair of keys is much larger than the safe key size for symmetric keys to ensure that N is not factorizable. For this reason, the potential attacks on RSA are small, its resistance to attacks depends on the infeasibility of factorizing N.

4. What is a distributed denial-of-service attack and how does it work?

Denial of service attack: Flooding a channel or other resource with messages in order to deny access for others.

OR

Distributed denial-of-service attacks target websites and online services. The aim is to overwhelm them with more traffic than the server or network can accommodate. The goal is to render the website or service inoperable. Flooding a channel or other resource with messages in order to deny access to others.

Here are the general steps that may be involved in a DDoS attack:

1. The attacker identifies a target, which can be a website, server, or network.
2. The attacker infects a large number of devices, such as personal computers or IoT devices, with malware that allows the attacker to remotely control them. These infected devices are referred to as "bots" or "zombies."
3. The attacker uses the bots to launch a coordinated flood of requests to the target, overwhelming its servers and preventing legitimate traffic from getting through.
4. The attack traffic may come from a single source or multiple sources, making it difficult to block or filter.
5. The target's servers and network infrastructure become overwhelmed and unable to respond to legitimate requests, causing the targeted website or service to become unavailable to users.
6. The attack continues until the attacker chooses to stop it or the target takes measures to block or mitigate the attack traffic.
7. It's important to note that there are different types of DDoS attacks and the exact method used can vary depending on the type of attack and the target. Additionally, the attackers are using new techniques like using botnets, amplification, spoofing, etc to make their attack more sophisticated.

---

5. What is the goal of security? List the three broad classes of security threats?

Goal: Use symmetric cryptography, public key cryptography, random numbers, and hash functions to enable exchange encryption keys, provide secure communication, and ensure message.

Classes of threats

1. **Leakage:** Refers to the acquisition of information by unauthorized recipients.
2. **Tampering:** Refers to the unauthorized alteration of information.
3. **Vandalism:** Refers to interference with the proper operation of a system without gain to the perpetrator.

6. What is cryptography? What is the use of it?

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it.

Applications Of Cryptography:

1. Computer passwords
2. Digital Currencies
3. Secure web browsing
4. Electronic Signatures
5. Authentication
6. Cryptocurrencies
7. End-to-end encryption

Uses of cryptography

1. Secrecy and integrity

- Cryptography is used to maintain the secrecy and integrity of information whenever it is exposed to potential attacks.
- It maintains the secrecy of the encrypted message as long as the decryption key is not compromised.
- It also maintains the integrity of the encrypted information, provided that some redundant information such as a checksum is included and checked.

2. Authentication

- Cryptography is used in support of mechanisms for authenticating communication between pairs of principals.
- Key is known only to two parties.

3. Digital signatures:

- Cryptography can be used to provide digital signatures, which are a way of verifying the authenticity of a message or document. A digital signature is created by encrypting a message with the sender's private key, and can be verified by decrypting it with the corresponding public key. This ensures that the message was sent by the holder of the private key, and that the message has not been tampered with in transit. Digital signatures are commonly used in electronic commerce and other applications to provide non-repudiation.

7. Write a note on digital signature?

A digital signature is a method of verifying the authenticity and integrity of a digital document or message. It is created by encrypting a message or document using the private key of the sender, and can be verified by decrypting it using the corresponding public key. A digital signature provides several benefits:

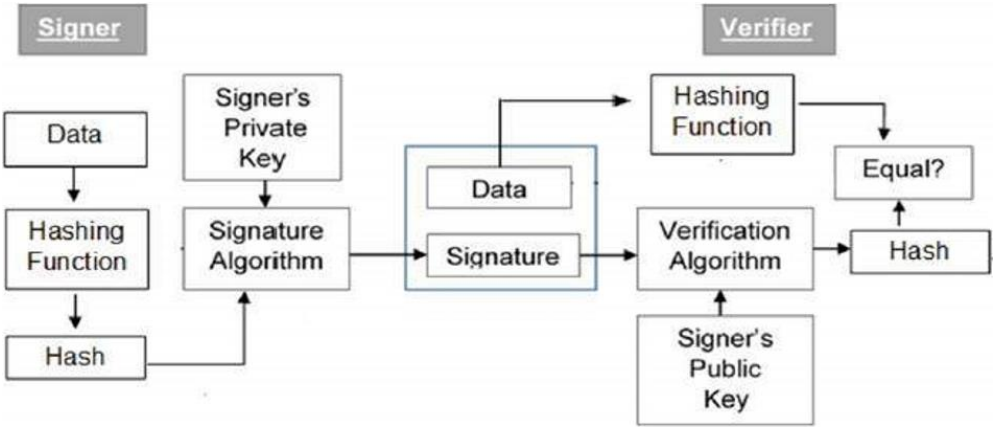
1. **Authentication:** A digital signature verifies that the message or document was sent by the holder of the private key, and not by an imposter.
2. **Integrity:** A digital signature ensures that the message or document has not been tampered with in transit, as any changes made to the message will result in a different digital signature.
3. **Non-repudiation:** A digital signature provides evidence that the sender cannot later deny having sent the message or document.

Digital signatures are commonly used in a variety of applications such as electronic commerce, email, and software distribution.

There are two main types of digital signatures:

1. **Symmetric digital signature:** One key is used for signing and verification.
2. **Asymmetric digital signature:** Public key is used for signing and private key is used for verification

Model of Digital Signature



UNIT – 4

Time and Global States

1. **Define following terms**

- **Physical clock**

Every computer has its own physical clock. These clocks are electronic devices that count oscillations occurring in a crystal that vibrates at a specific frequency when electricity is applied, and typically divide this count and store the result in a counter register. Clock devices can be programmed to generate interrupts at regular intervals in a particular order. The speed of a computer processor is measured in clock speed.

- **Clock skew and clock drift**

Computer clocks, like any other clocks, tend not to be in perfect agreement. The instantaneous difference between the readings of any two clocks is called their skew. Also, the crystal-based clocks used in computers are, like any other clocks, subject to clock drift, which means that they count time at different rates, and so diverge. A clock's drift rate is the change in the offset between the clock and a nominal perfect reference clock per unit of time measured by the reference clock.

- **Clock Skew** = Relative Difference in clock values of two processes.
- **Clock Drift** = Relative Difference in clock frequencies (rates) of two processes.

OR

Clock skew

- The difference between the times on two clocks (at any instant)
- Computer clocks use crystal-based clocks that are subject to physical variations

Clock drift

- They count time at different rates and so diverge (frequencies of oscillation differ)

- **Coordinated Universal Time**

Computer clocks can be synchronized to external sources of highly accurate time. The most accurate physical clocks use atomic clocks and are used as the standard for elapsed real-time, known as International Atomic Time. Coordinated Universal Time also known as UTC is an international standard for timekeeping. It is based on atomic time. UTC signals are synchronized and broadcast regularly from land-based radio stations and satellites covering many parts of the world.

2. Explain different modes of synchronizing a physical clocks.

In order to know at what time of day events occur at the processes in our distributed system – for example, for accountancy purposes – it is necessary to synchronize the processes' clocks, C_i, with an authoritative, external source of time. This is external synchronization. And if the clocks C_i are synchronized with one another to a known degree of accuracy, then we can measure the interval between two events occurring at different computers by appealing to their local clocks, even though they are not necessarily synchronized to an external source of time. This is internal they are not necessarily synchronized to an external source of time. This is internal of real time I (capital i):

Two models of synchronization

1. **External synchronization:** a computer's clock Ci is synchronized with an external authoritative time source S, so that:

$$|S(t) - C_i(t)| < D \text{ for } i = 1, 2, \ldots N \text{ over an interval, } I \text{ of real time}$$

The clocks C_i are accurate to within the bound *D*.

2. **Internal synchronization:** the clocks of a pair of computers are synchronized with one another so that:

$$| C_i(t) - C_j(t)| < D \text{ for } i = 1, 2, \ldots N \text{ over an interval, I of real time}$$

The clocks C_i and C_j **agree** within the bound D.

Internally synchronized clocks are not necessarily externally synchronized, as they may drift collectively

- if the set of processes P is synchronized externally within a bound D, it is also internally synchronized within bound 2D (worst case polarity)

3. Explain Cristian's method for synchronizing clocks.

Cristian's Algorithm is a clock synchronization algorithm is used to synchronize time with a time server by client processes.

Algorithm:

1. The process on the client machine sends the request for fetching clock time (time at the server) to the Clock Server at time $\mathbf{T_0}$.
2. The Clock Server listens to the request made by the client process and returns the response in form of clock server time ($\mathbf{T_1}$).
3. The client process fetches the response from the Clock Server at time $\mathbf{T_1}$ and calculates the synchronized client clock time using the formula given below,

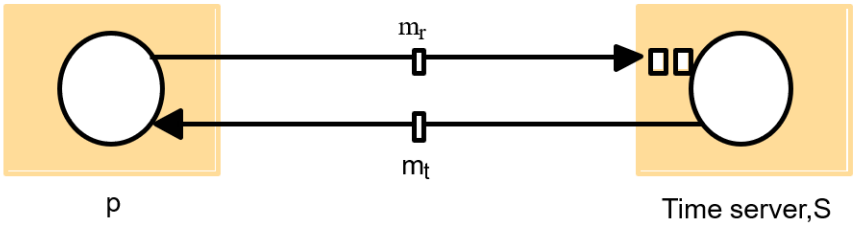
$$\mathbf{T_{CLIENT} = T_{SERVER} + (T_1 - T_0)/2}$$

OR

A time server S receives signals from a UTC source

- Process p requests time in m_r and receives t in m_t from S
- p sets its clock to $t + T_{round}/2$
- Accuracy $\pm(T_{round}/2 - min)$:
  - because the earliest time S puts t in message m_t is min after p sent m_r
  - the latest time was min before m_t arrived at p
  - the time by S's clock when m_t arrives is in the range $[t+min, t + T_{round} - min]$
  - the width of the range is $T_{round} + 2min$

T_{round} is the round trip time recorded by p
min is an estimated minimum round trip time



Cristian suggested the use of a time server, connected to a device that receives signals from a source of UTC, to synchronise computers externally. Upon request, the server process S supplies the time according to its clock as shown in the figure. Cristian Observed that while there is no upper bound on message transmission delays in an asynchronous system, the round-trip times for messages exchanged between pairs of processes are often reasonably short - a small fraction of a second. He describes the algorithm as *probabilistic*: the method achieves synchronisation only if the observed round-trip times between client and server are sufficiently short compared with the required accuracy. A process p request the time in a message m_r, and receives the time value t in a message m_t. Process p records the total round trip time T_{round} taken to send the request m_r and receive the reply m_t. It can measure this time with reasonable accuracy if it's rate of clock drift is small.

** Round-trip time (RTT) is the duration measured in milliseconds, from when a browser sends a request to when it receives a response from a server.

**Clock synchronization is the mechanism to synchronize the time of all the computers in the distributed environments or system.

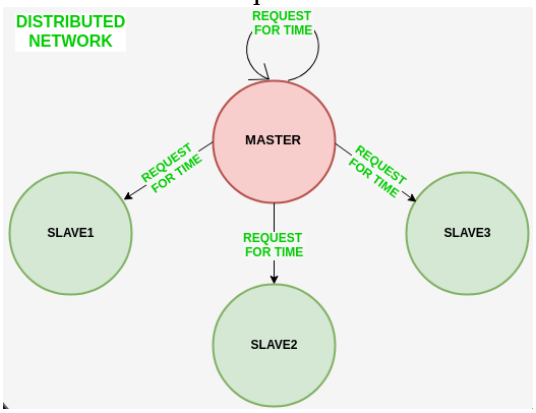
4. Explain Berkeley algorithm for internal synchronization.

Berkeley algorithm - a method of clock synchronization in distributed computing which assumes no machine has an accurate time source.

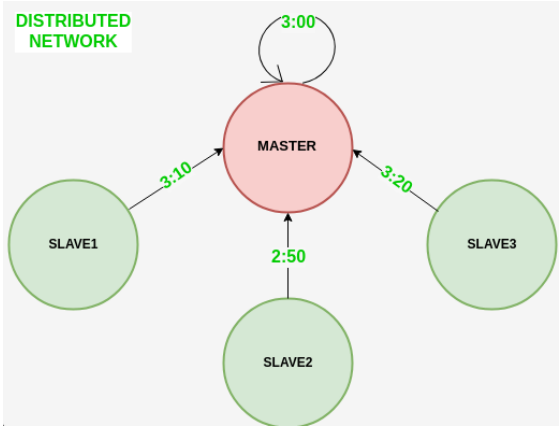
Algorithm

1. An individual node is chosen as the master node from a pool node in the network. This node is the main node in the network which acts as a master and the rest of the nodes act as slaves. The master node is chosen using an election process/leader election algorithm.
2. Master node periodically pings slaves nodes and fetches clock time at them using Cristian's algorithm.

The diagram below illustrates how the master sends requests to slave nodes.

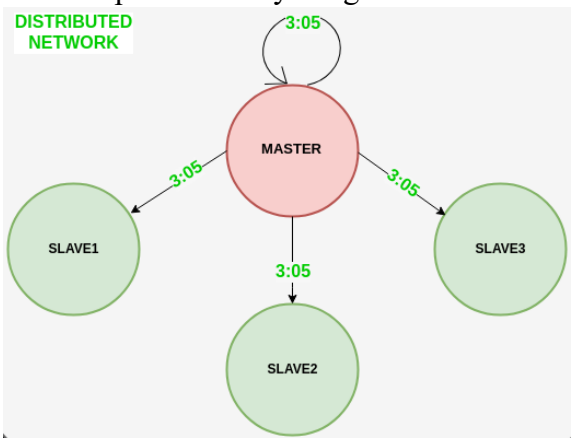


The diagram below illustrates how slave nodes send back time given by their system clock.

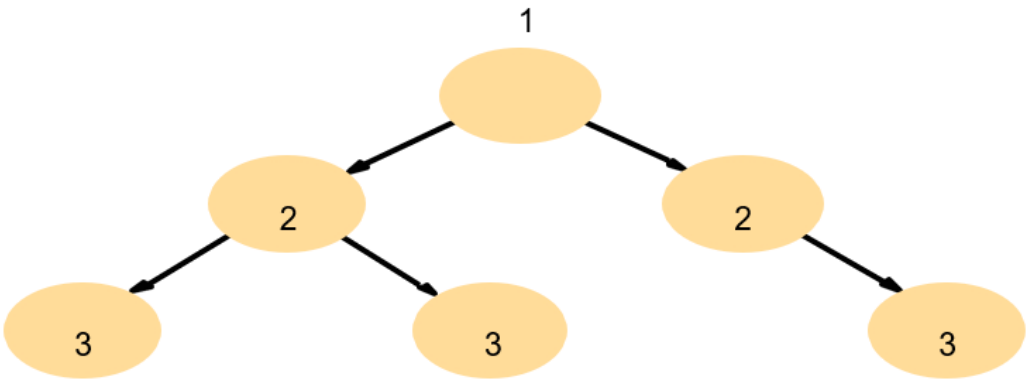


3. Master node calculates the average time difference between all the clock times received and the clock time given by the master's system clock itself. This average time difference is added to the current time at the master's system clock and broadcasted over the network.

The diagram below illustrates the last step of Berkeley's algorithm.



5. **With the neat diagram, explain the concept of synchronization subnet in an NTP implementation.**



An example of synchronization subnet in an NTP implementation

- The NTP service is provided by a network of servers located across the Internet. Primary servers are connected directly to a time source such as a radio clock receiving UTC; secondary servers are synchronized, ultimately, with primary servers.
- The servers are connected in a logical hierarchy called a synchronization subnet whose levels are called strata.
- Primary servers occupy stratum 1: they are at the root. Stratum 2 servers are secondary servers that are synchronized directly with the primary servers; stratum 3 servers are synchronized with stratum 2 servers, and so on.
- The lowest-level (leaf) servers execute in users' workstations. The clocks belonging to servers with high stratum numbers are liable to be less accurate than those with low stratum numbers, because errors are introduced at each level of synchronization.
- NTP also takes into account the total message round-trip delays to the root in assessing the quality of timekeeping data held by a particular server. The synchronization subnet can reconfigure as servers become unreachable or failures occur.
- If, for example, a primary server's UTC source fails, then it can become a stratum 2 secondary server.
- If a secondary server's normal source of synchronization fails or becomes unreachable, then it may synchronize with another server.
- NTP servers synchronize with one another in one of three modes: multicast, procedure-call and symmetric mode. Multicast mode is intended for use on a high-speed LAN.
- One or more servers periodically multicasts the time to the servers running in other computers connected by the LAN, which set their clocks assuming a small delay.
- This mode can achieve only relatively low accuracies, but ones that nonetheless are considered sufficient for many purposes.

6. Discuss different modes of NTP server synchronization.

NTP servers synchronize with one another in one another in one of the 3 modules:

1. Multicast mode
2. Procedure-call mode
3. Symmetric mode

Multicast mode

- It is intended for use on a high speed LAN.
- One or more servers periodically multicasts the time to the servers running in other computers connected by the LAN, which set their clocks assuming a small delay.
- This mode can achieve only relatively low accuracies, but one that nonetheless are considered sufficient for many purposes.

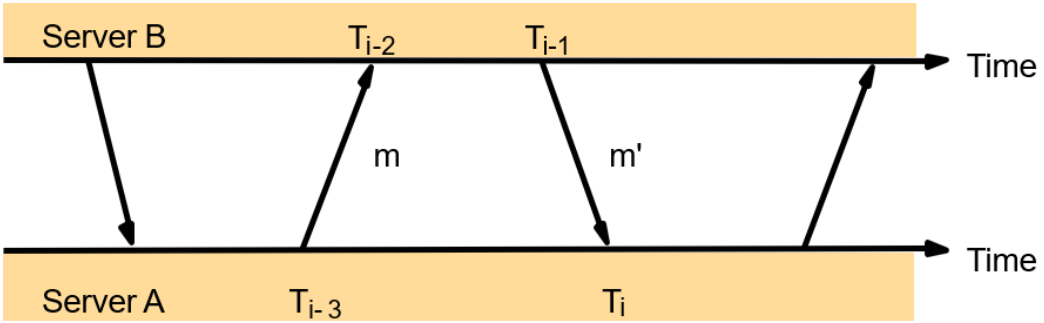
Procedure call

- A server similar to the operation of Cristian's algorithm.
- In this mode, one server accepts requests from other computers, which it processes by replying with its timestamp (current clock reading).
- This mode is suitable where higher accuracies are required than can be achieved with multicast, or where multicast is not supported in hardware.
- For E.g., file servers on the same or a neighboring LAN that need to keep accurate timing information for file accesses could contact a local server in procedure-call mode.

Symmetric

- It is intended for use by the servers that supply time information in LANs and by the higher levels (lower strata) of the synchronization subnet, where the highest accuracies are to be achieved.
- A pair of servers operating in symmetric mode exchange messages are to be achieved.
- Timing data are retained as part of an association between the servers that is maintained in order to improve to the accuracy of their synchronization over time.

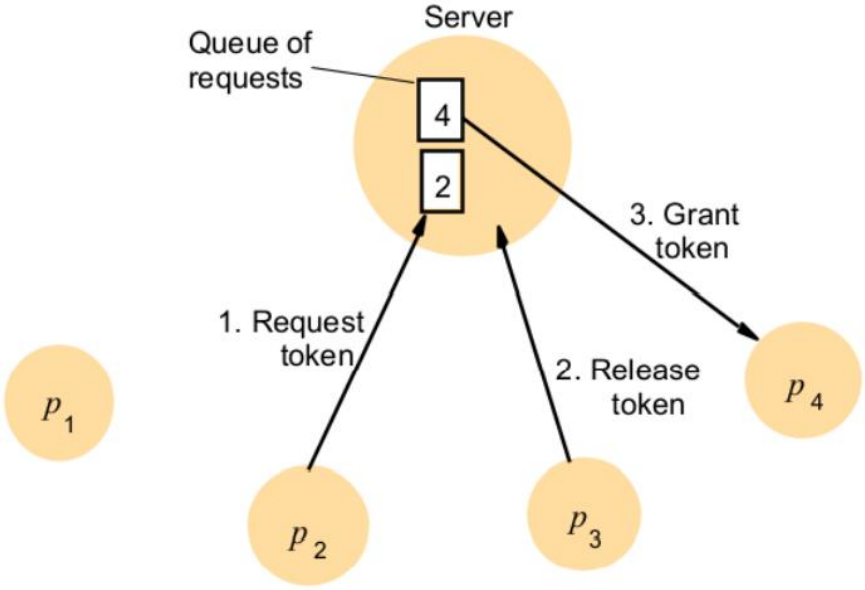
7. Explain with the neat diagram how messages are exchanged between a pair of NTP peers.



- In all modes, messages are delivered unreliably, using the standard UDP Internet transport protocol.
- In procedure-call mode and symmetric mode, processes exchange pairs of messages. Each message bears timestamps of recent message events:
  - The local times when the previous NTP message between the pair was sent and received.
  - The local time when the current message was transmitted.
- The recipient of the NTP message notes the local time when it receives the message. The four times $\mathbf{T_{i-3}}$, $\mathbf{T_{i-2}}$, $\mathbf{T_{i-1}}$ and $\mathbf{T_i}$ are shown in figure for the messages **m** and **m'** sent between servers A and B.
- In symmetric mode there can be a nonnegligible delay between the arrival of one message and the dispatch of the next.
- Messages may be lost, but the three timestamps carried by each message are nonetheless valid.

Co-ordination & Agreement

1. **Explain with a neat diagram Central Server Algorithm.**



The simplest way to achieve mutual exclusion is to employ a server that grants permission to enter the critical section. The figure shows the use of this server. To enter a critical section, a process sends a request message to the server and awaits a reply from it. Conceptually, the reply constitutes a token specifying permission to enter the critical section. If no other process has the token at the time of the request, then the server replies immediately, granting the token. If the token is currently held by another process, then the server does not reply, but queues the request. When a process exits the critical section, it sends a message to the server, giving it back the token. If the queue of waiting processes is not empty, then the server chooses the oldest entry in the queue, removes it and replies to the corresponding process.

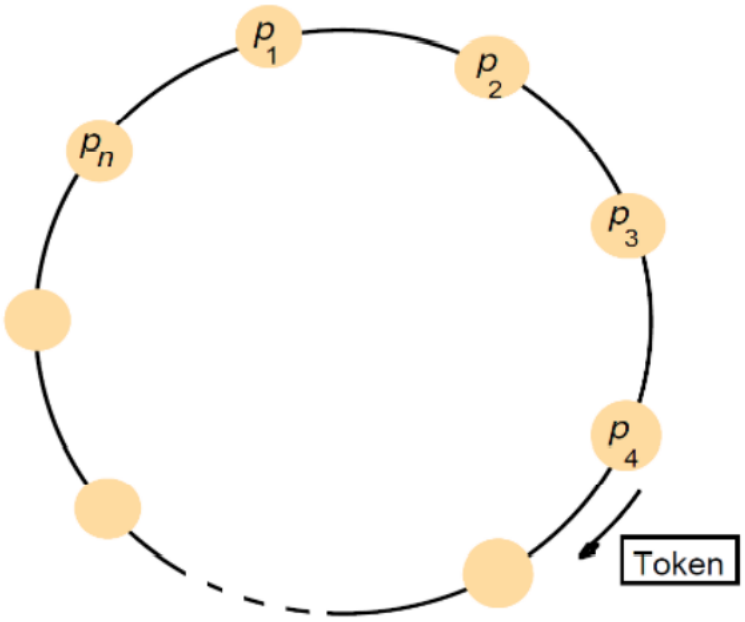
****Critical section** - typically used when a multi-threaded program must update multiple related variables without a separate thread making conflicting changes to that data.

OR

When more than one processes access the same code segment.

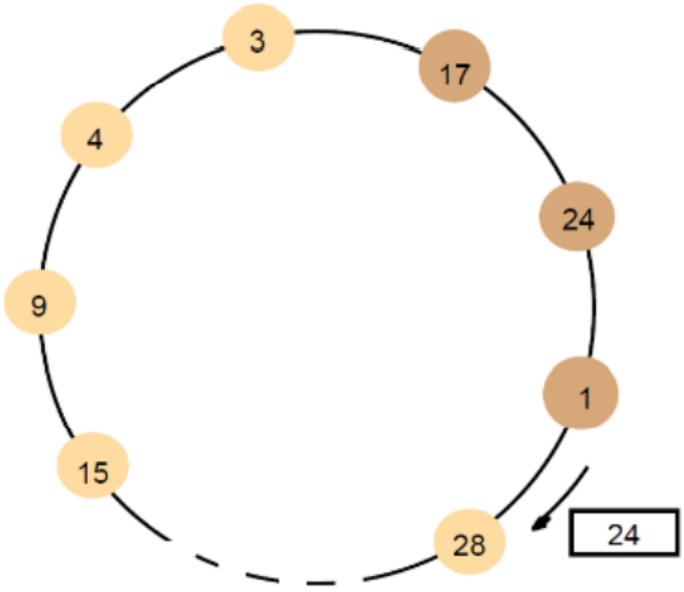
---

2. With neat diagram explain Ring based Algorithm w.r.t. mutual exclusion



One of the simplest ways to arrange mutual exclusion between the N processes without requiring an additional process is to arrange them in a logical ring. This requires only that each process p_i has a communication channel to the next process in the ring, $p_{(i+1) mod N}$. The idea is that exclusion is conferred by obtaining a token in the form of a message passed from process to process in a single direction, ex: clockwise or around the ring. The ring topology may be unrelated to the physical interconnections between the underlying computers. If a process does not require to enter the critical section when it receives the token, then it immediately forwards the token to its neighbour. A process that requires the token waits until it receives it, but retain it. To exit the critical section, the process sends the token on to its neighbour.

3. With neat diagram explain Ring based Election Algorithm.



Note: The election was started by process 17.
The highest process identifier encountered so far is 24.
Participant processes are shown in a darker colour

The algorithm of Chang and Roberts is suitable for a collection of processes arranged in a logical ring. Each process p_i has a communication channel to the next process in the ring, $p_{(i+1) mod N}$, and all the messages are sent clockwise around the ring. We assume that no failures occur, and that the system is asynchronous. The goal of this algorithm is to elect a single process called the coordinator, which is the process with the largest identifier.

Initially, every process is marked as non-participant in an election. Any process can begin an election. It proceeds by marking itself as a participant, placing its identifier in an election message and sending it to its clockwise neighbour.

When a process receives an election message, it compares the identifier in the message with its own. If the arrived identifier is greater, then it forwards the message to its neighbour. If the arrived identifier is smaller and the receiver is not a participant, then its substitute its own identifier in the message and forwards it. If, however, the received identifier is that of the receiver itself, then this process's identifier must be the greatest, and it becomes the coordinator. The coordinator marks itself as a non-participant once more and sends an elected message to its neighbour, announcing its election and enclosing its identity.

UNIT – 5

1. Define cloud computing and attributes for delivering computing services.

Cloud computing is providing different IT services to customers over the Internet. It is the ability to deliver computing service over the Internet to the end-users on-demand or on a pay-as-you-go basis.

Attributes

- Cloud computing uses Internet technologies to offer elastic services. The term elastic computing refers to the ability to dynamically acquire computing resources and support a variable workload. A cloud service provider maintains a massive infrastructure to support elastic services.
- The resources used for these services can be metered and the users can be charged only for the resources they use.
- Maintenance and security are ensured by service providers.
- Economy of scale allows service providers to operate more efficiently due to specialization and centralization.
- Cloud computing is cost-effective due to resource multiplexing; lower costs for the service provider are passed on to the cloud users.
- The application data is stored closer to the site where it is used in a device and location independent manner; potentially, this data storage strategy increases reliability and security and, at the same time, it lowers communication costs.

2. **Discuss network centric computing and network centric content.**

Network-centric computing

- Focuses on the network as the central point of control and management
- Allows for easy access and sharing of resources (such as storage and computing power) across multiple users and devices
- Often used in cloud computing environments to improve scalability and performance
- Allows for better resource utilization and cost-efficiency
- Can improve security by centralizing access control and monitoring
- Typical examples of large-scale network-centric systems are the **World-Wide Web and Computational Grids.**

Network-centric content

- Utilizes network-based resources (such as cloud storage and CDNs) to store, manage, and distribute content
- Allows for easy access to content from any location and device
- Improves performance and scalability by distributing content across multiple servers and locations
- Can reduce costs associated with storage and distribution of large amounts of data.
- Allows for dynamic and adaptive content delivery based on network conditions and user preferences.
- Network-centric content is used in e-commerce to distribute and manage large amount of data such as images, videos, and product descriptions. Network-centric computing allows for efficient management of resources and infrastructure for online stores.

Network-centric computing and content

- **Data-intensive:** large scale simulations in science and engineering require large volumes of data. Multimedia streaming transfers large volume of data.
- **Network-intensive:** transferring large volumes of data requires high bandwidth networks.
- Low-latency networks for data streaming, parallel computing, computation steering.
- The systems are accessed using *thin clients* running on systems with limited resources, e.g., wireless devices such as smart phones and tablets.
- The infrastructure should support some form of *workflow management*.

3. Explain different types of clouds with examples.

1. Public Cloud

The infrastructure is made available to the general public or a large industry group and is owned by the organization selling cloud services.

2. Private Cloud

The infrastructure is operated solely for an organization.

3. Community Cloud

The infrastructure is shared by several organizations and supports a community that has shared concerns.

4. Hybrid Cloud

Composition of two or more clouds (public, private, or community) as unique entities but bound by standardized technology that enables data and application portability.

Private Cloud

- Private cloud refers to a cloud deployment model operated exclusively to a single organization. It provides computing services to a private internal network and selected users, instead of the public in general.
- **E.g.:** HP Data Centers, Elastra-private cloud, Ubuntu, etc.

Public Cloud

- In this, the business rents the services that are required and pays for what is utilized on-demand. The resources are owned, maintained & operated by a third-party cloud service provider, and delivered over the internet.
- **E.g.:** AWS, GCP, Microsoft Azure, etc.

Community Cloud

- Community clouds are distributed systems created by integrating the services of different clouds to address the specific needs of an industry, a community, or a business sector. The cloud may be managed by an organization or a third party.
- **E.g.:** Our government organization within India may share computing infrastructure in the cloud to manage data.

Hybrid Cloud

- A hybrid cloud is a heterogeneous distributed system formed by combining facilities of public cloud and private cloud. For this reason, they are also called heterogeneous clouds.
- A major drawback of private deployments is the inability to scale on-demand and efficiently address peak loads. Here public clouds are needed. Hence, a hybrid cloud takes advantage of both public and private clouds.
- **E.g.:** Implementing database using on-premise private servers and using third party cloud service providers like AWS as load balancing or computing solutions.

5. **Discuss the success and failure of cloud computing.**

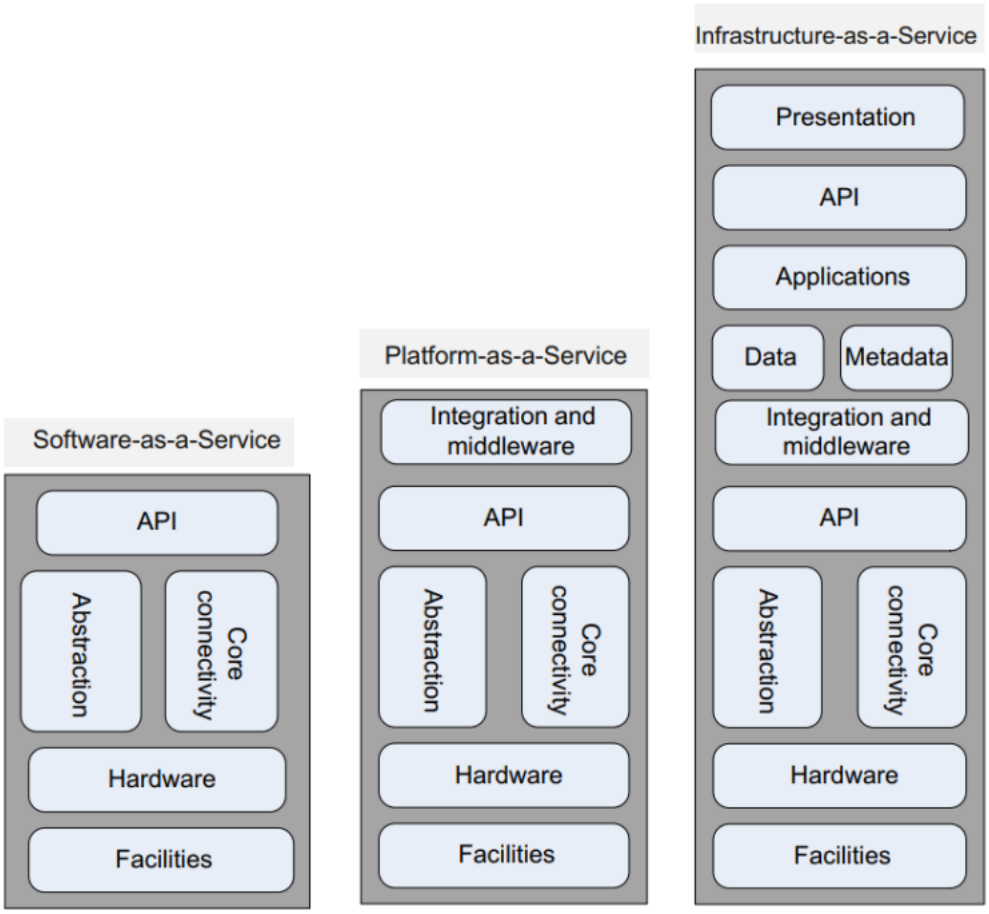
Advantages

1. **Flexibility:** Remote cloud servers offer almost unlimited bandwidth and storage space, which allows businesses to instantly scale up and down their capacities to support growth and cope when website traffic increases. This removes the need to purchase, configure and install equipment on-site.
2. **Business Continuity:** By investing in cloud computing, businesses can guarantee reliable disaster recovery and backup solutions without the hassle of setting them up on a physical device.
3. **Cost Efficiency:** The most significant advantage of cloud computing is the IT operational cost savings. Using remote servers removes the need for in-house storage equipment and application requirements, as well as overhead costs such as software updates, management, and data storage.
4. **Scalability and Performance:** Cloud technology is designed to be scaled to meet a business's changing IT requirements. As a company grows, it is inevitable that more storage space and bandwidth will be required to cope with increasing traffic to the website. Cloud servers can be deployed automatically to help businesses scale up and down and ensure optimum performance under heavy loads.
5. **Automatic Software Updates:** Many cloud service providers offer regular system updates to ensure IT requirements are consistently met. They ensure round-the-clock maintenance of cloud servers – including security updates.

Disadvantages

1. **Requires good speed internet with good bandwidth:** To access your cloud services, you need to have a good internet connection always with good bandwidth to upload or download files to/from the cloud.
2. **Downtime:** Since the cloud requires high internet speed and good bandwidth, there is always a possibility of service outage, which can result in business downtime. Today, no business can afford revenue or business loss due to downtime or slow down from an interruption in critical business processes.
3. **Limited control of infrastructure:** Since you are not the owner of the infrastructure of the cloud, hence you don't have any control or have limited access to the cloud infra.
4. **Restricted or limited flexibility:** The cloud provides a huge list of services, but consuming them comes with a lot of restrictions and limited flexibility for your applications or developments. Also, platform dependency or 'vendor lock-in' can sometimes make it difficult for you to migrate from one provider to another.
5. **Ongoing costs:** Although you save your cost of spending on whole infrastructure and its management, on the cloud, you need to keep paying for services as long as you use them. But in traditional methods, you only need to invest once.

6. **With the neat diagram explain cloud computing delivery models and services.**



Cloud delivery models

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

1. Software-as-a-Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.

2. Platform-as-a-Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

3. Infrastructure-as-a-Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources. Services offered by this delivery model include: server hosting, web servers, storage, computing hardware, operating systems, virtual instances, load balancing, Internet access, and bandwidth provisioning.

---

7. Discuss ethical issues encountered in cloud computing.

Issues are as follows:

1. **Privacy:** Cloud computing involves storing personal data on remote servers, which can raise concerns about data privacy and security. Users may be concerned about the access and control that cloud providers have over their personal data.
2. **Data security:** Cloud computing also presents security risks, such as the potential for data breaches and unauthorized access to personal information.
3. **Jurisdiction:** Cloud providers may store data in different countries, which can create legal and ethical issues around data sovereignty and jurisdiction.
4. **Transparency:** Cloud providers may not be transparent about how they collect, use, and share user data, which can raise concerns about informed consent and user control over their personal information.
5. **Interoperability:** Some cloud providers may use proprietary standards and lock-in customers, making it difficult or impossible to move data and applications to other providers.
6. **Dependence:** Heavy dependency on cloud providers can lead to business continuity issues, if provider goes out of business or decide to stop providing certain service.
7. **Environmental Impact:** Cloud computing requires significant energy consumption and infrastructure, which can have a negative impact on the environment.
8. **Digital divide:** Limited access to cloud computing services can exacerbate existing social and economic inequalities, creating a "digital divide" between those who have access to these services and those who do not.

---

8. Discuss the challenges faced by cloud computing

1. **Security:** One of the biggest challenges facing cloud computing is ensuring the security of data stored and processed on remote servers. This includes protecting against data breaches, unauthorized access, and other security threats.
2. **Data privacy:** Another challenge is ensuring that personal data is protected and kept private. This includes ensuring that cloud providers are transparent about how they collect, use, and share user data.
3. **Interoperability:** As different cloud providers use different technologies and standards; it can be difficult for organizations to move data and applications between different cloud environments.
4. **Compliance:** Organizations may have to comply with various laws and regulations regarding data storage, processing, and sharing. It can be challenging to ensure compliance with these regulations in a cloud computing environment.
5. **Reliability:** Cloud providers must ensure that their services are highly available and reliable, as downtime can have a significant impact on businesses and users.
6. **Governance:** Cloud providers must ensure that they have proper governance in place to manage the resources and services they offer.
7. **Cost:** Organizations must consider the cost of cloud computing, including the cost of data storage and processing, as well as the cost of migrating and managing applications and data in the cloud.
8. **Performance:** Organizations may experience performance issues, such as latency and bandwidth limitations, when using cloud computing services, particularly when accessing data from remote locations.
9. **Dependence:** Organizations may become too dependent on cloud providers, creating business continuity issues if the provider goes out of business or decide to stop providing certain service.

---

9. Explain cloud vulnerabilities.

- Clouds are affected by malicious attacks and failures of the infrastructure, e.g., power failures.
- Such events can affect the Internet domain name servers and prevent access to a cloud or can directly affect the clouds:
  - in 2004 an attack at Akamai caused a domain name outage and a major blackout that affected Google, Yahoo, and other sites.
  - in 2009, Google was the target of a denial of service attack which took down Google News and Gmail for several days.
  - in 2012 lightning caused a prolonged down time at Amazon.
  - ***Check ppt***